# IEEE Standard for a High Performance Serial Bus

Sponsor

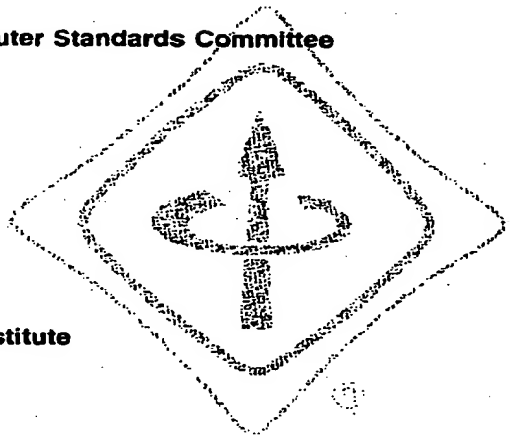
**Microprocessor and Microcomputer Standards Committee of the IEEE Computer Society**

Approved 12 December 1995

**IEEE Standards Board**

Approved 22 July 1996

**American National Standards Institute**



**Abstract:** A high-speed serial bus that integrates well with most IEEE standard 32-bit and 64-bit parallel buses, as well as such nonbus interconnects as the IEEE Std 1596-1992, Scalable Coherent Interface, is specified. It is intended to provide a low-cost interconnect between cards on the same backplane, cards on other backplanes, and external peripherals. This standard follows the IEEE Std 1212-1991 Command and Status Register (CSR) architecture.

**Keywords:** backplane, bus, computers, high-speed serial bus, interconnect, parallel buses

---

The Institute of Electrical And Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017-2394, USA



ISBN 1-55937-583-3

**IEEE Standards** documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE that have expressed an interest in participating in the development of the standard.

Use of an IEEE Standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of all concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
USA

Note: Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying all patents for which a license may be required by an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

# Annex E

(informative)

# Cable operation and implementation examples

## E.1 Timing formulas for cable environment gap control

When configuring the cable topology, there are three primary ways to increase Serial Bus performance. All methods are dependent on the topology implementor and the ability of the Bus Manager to optimize for that topology.

a) Reduce the number of hops
b) Put nodes of the same speed capability next to one another
c) Optimize the gap_count for the Bus Manager for the current system configuration

Items a) and b) are only dependent on the topology implementor and are beyond the control of the bus management software. The third item is controlled by the bus management software. Setting the gap_count to its optimum value should be done to increase the efficiency of the Serial Bus. In the following paragraphs the purpose and value for the gap_count are explained.

The gap_count ensures that all nodes on the bus see the appropriate gap times. For example, in a cable topology where there are 16 hops from one end of the Serial Bus to the other, the signal propagation delay is significant. With a gap_count of 1, a subaction gap time is just 0.44 μs and an arb reset gap is 0.84 μs. If the propagation delay is greater than the arb reset gap time minus the subaction gap time (0.40 μs), then the node that released the bus (beginning its gap timer) will see an arb reset gap before the node that is 16 hops away sees a subaction gap. This can cause the asynchronous bus bandwidth to be allocated unfairly. The following example demonstrates the importance of optimizing the gap_count and reducing the number of hops to increase the Serial Bus performance.

This example assumes the following:

— All timing constants and formulas are from tables 4-32, 4-33, and 4-34.
— S100 transfer rate
— Asynchronous transfer
— 512 bytes data, 24 bytes overhead
— Cable velocity of propagation of 5.05 ns/m as specified in 4.2.1.4.3
— Cable assemblies of 4.5 m as specified in 4.2.1.2.2

The last two assumptions result in a single-hop cable delay of

Cable Delay = 4.5 m · 0.00505 μs = 0.022725 μs

Sending of a packet can be divided into four phases:

1) Arbitration (Arb Phase)
2) Data transfer (Data Phase)
3) Acknowledgment (Ack Phase)
4) Between packet gap times (Gap Phase)

### E.1.1 Arbitration phase

Arbitration delay is the delay that all nodes have to wait before starting arbitration. This value varies with the gap_count.

arb_delay = gap_count · 4/BASE_RATE

**Table E-1 — Arbitration delays**

| gap_count | Minimum arb_delay (µs) | Maximum arb_delay (µs) |
|---|---|---|
| 1 | 0.04068 | 0.04094 |
| 10 | 0.40686 | 0.40942 |
| 20 | 0.81371 | 0.81388 |
| 30 | 1.22057 | 1.22082 |
| 40 | 1.62743 | 1.62777 |
| 50 | 2.03429 | 2.03471 |
| 63 | 2.56321 | 2.56373 |

Speed signaling and the data prefix are done in parallel on the bus. ARB_SPEED_SIGNAL_START is the time that speed signaling leads the data prefix. In the minimum case, speed signaling is more time consuming than the data prefix time. From table 4-32:

-0.02 µs < ARB_SPEED_SIGNAL_START

0.10 µs < SPEED_SIGNAL_LENGTH < 0.12 µs

0.04 µs < DATA_PREFIX_TIME < 0.16 µs

MAX_DATA_PREFIX_DELAY is the maximum delay between the data prefix arriving at a node and the same data prefix being sent by that node. This delay is multiplied by the number of hops in the topology to find the worst-case system delay. The total arbitration phase time varies with the gap_count and the number of hops (N) in the bus topology. The following formulas were used to calculate this time:

*Minimum Case:*

Arb Phase Time =
(Cable Delay · N) + arb_delay + SPEED_SIGNAL_LENGTH

*Maximum Case:*

Arb Phase Time =
(Cable Delay · N) + arb_delay - (-ARB_SPEED_SIGNAL_START) +
DATA_PREFIX_TIME + (N · MAX_DATA_PREFIX_DELAY)

**Table E-2 — Total arbitration phase time (µs)**

| gap_count | Minimum | Maximum 1 hop | Maximum 8 hops | Maximum 16 hops |
|---|---|---|---|---|
| 1 | 0.1434 | 1.8558 | 3.0119 | 4.3332 |
| 10 | 0.5096 | 2.2221 | 3.3782 | 4.6994 |
| 20 | 0.9164 | 2.6290 | 3.7851 | 5.1064 |
| 30 | 1.3233 | 3.0360 | 4.1921 | 5.5133 |
| 40 | 1.7302 | 3.4429 | 4.5990 | 5.9202 |
| 50 | 2.1370 | 3.8499 | 5.0060 | 6.3272 |
| 63 | 2.6659 | 4.3789 | 5.5350 | 6.8562 |

### E.1.2 Data transfer phase

The packet transmission time is based on an asynchronous block write request with a data packet size of 512 bytes and 24 bytes of overhead.

Packet Transmission Time =

(512 + 24) · 8 / Base Rate

giving

43.6153 µs < Packet Transmission Time < 43.6242 µs

MAX_PHY_DATA_DELAY is the maximum delay between the data arriving at a node and the same data being sent by that node. This delay is multiplied by the number of hops in the topology to find the worst-case system delay.

Data Transfer Phase Time =
(Cable Delay · N) + Pkt Transmission Time + MAX_PHY_DATA_DELAY · N

The results from this formula can be found in table E-3.

**Table E-3 — Total data transfer phase time**

| N = number of hops | Mininum (µs) | Maximum (µs) |
|---|---|---|
| 1 | 43.638 | 43.789 |
| 8 | 43.797 | 44.942 |
| 16 | 43.979 | 46.260 |

### E.1.3 Acknowledgment phase

The Acknowledgment phase consists of the acknowledge gap time, ACK_RESPONSE_TIME, and the acknowledge transmission time. The acknowledge gap time is the time between the end of a packet and the beginning of the acknowledge:

0.04 µs < acknowledge_gap_time < 0.05 µs

The ACK_RESPONSE_TIME is the time between reporting the end of a packet (DATA_END) and the acknowledging link layer requesting arbitration to send the acknowledgment.

0.05 µs < ACK_RESPONSE_TIME < 0.17 µs

There are 8 bits in an acknowledge, so the ack transmission time is 8 / BASE_RATE:

0.04 µs < Ack Transmission Time < 0.05 µs

The total acknowledge phase time is

Ack Phase Time =
ack gap + (ACK_RESPONSE_TIME - acknowledge_gap_time) + (Cable Delay · N) +
Ack Transmission Time + MAX_PHY_DATA_DELAY · N

The results from this formula can be found in table E-4.

**Table E-4 — Total acknowledge phase time**

| N = number of hops | Mininum (µs) | Maximum (µs) |
|---|---|---|
| 1 | 0.1541 | 0.4161 |
| 8 | 0.3132 | 1.5692 |
| 16 | 0.4950 | 2.8870 |

## E.1.4 Between packet gap times

In this phase, the time required for the subaction and arb reset gaps is determined for various value of gap_count.

(27 + gap_count · 16)/BASE_RATE ≤ subaction gap ≤ (29 + gap_count · 16)/BASE_RATE
(51 + gap_count · 32)/BASE_RATE ≤ arb reset gap ≤ (53 + gap_count · 32)/BASE_RATE

The total time from the subaction gap through the arb reset gap for an asynchronous write request with 512 bytes of data and 24 bytes of header and the corresponding acknowledge is listed in table E-5. The total time was calculated using the following formula:

Total time =
Arb Phase + Data Transfer Phase + Ack Phase + subaction gap + arb reset gap

**Table E-5 — Total time from subaction gap through arb reset gap**

| gap_count | Minimum (µs) | Maximum for 1 hop (µs) | Maximum for 8 hops (µs) | Maximum for 16 hops (µs) |
|---|---|---|---|---|
| 1 | 45.21720 | 47.40620 | 51.02751 | 55.16615 |
| 10 | 50.18198 | 52.16743 | 55.78874 | 59.92738 |
| 20 | 55.26663 | 57.45768 | 61.07899 | 65.21763 |
| 30 | 60.55580 | 63.24211 | 66.36924 | 70.50788 |
| 40 | 65.84498 | 68.69709 | 71.65950 | 75.79813 |
| 50 | 71.13416 | 74.15206 | 76.94975 | 81.08839 |
| 63 | 78.01008 | 81.19412 | 83.82707 | 87.96571 |

Table E-5 illustrates the importance of setting the gap_count to an optimum value. To do this, it is important that the worst-case topology be realized for the number of hops in the system.

For example: A system has 16 hops, Node_1 is 16 hops away from Node_16, and Node_1 is root and cannot send another packet during this fairness interval. When Node_1 releases the bus (starting its gap timer), the gap propagates through the topology, reaching Node_16 some propagation delay time later (Prop1). Node_16 can arbitrate for the bus after a subaction gap plus arb_delay (note that the gap timer for Node 1 has advanced to subaction gap + arb_delay + Prop1). The arbitration signal for Node 16 now propagates through the topology, reaching Node_1 some propagation delay time later (Prop2). The total delay time seen by Node_1 is (subaction gap + arb_delay + Prop1 + Prop2). The total delay time has to be less than the arbitration reset gap; otherwise, Node_1 can arbitrate for the bus and win. Therefore, the gap_count needs to be set to value where the (arb reset gap - subaction gap > Prop1 + Prop2).

Total propagation delay = Prop1 + Prop2 = N · 2 · (Cable Delay + MAX_PHY_DATA_DELAY)

With the total propagation delay known, it is now possible to select the appropriate gap_count for the number of hops by using the following formula:

arb reset gap - subaction gap > total propagation delay

The resulting table of gap_count values is given in table E-6.

**Table E-6 — Calculated gap counts**

| Maximum number of hops | Total propagation delay (μs) | Gap_count |
|---|---|---|
| 1 | 0.3295 | 1 |
| 2 | 0.6589 | 4 |
| 3 | 0.9884 | 6 |
| 4 | 1.3178 | 9 |
| 5 | 1.6473 | 12 |
| 6 | 1.9767 | 14 |
| 7 | 2.3062 | 17 |
| 8 | 2.6356 | 20 |
| 9 | 2.9651 | 23 |
| 10 | 3.2945 | 25 |
| 11 | 3.6240 | 28 |
| 12 | 3.9534 | 31 |
| 13 | 4.2829 | 33 |
| 14 | 4.6123 | 36 |
| 15 | 4.9418 | 39 |
| 16 | 5.2712 | 42 |

## E.2 Cable environment jitter budget

Tables E-7 through E-9 give the jitter budget for the three cable PHY data rates. These can be used to compute the jitter margin for each data rate using the following formula:

(Bit cell time – (Data jitter + Strobe jitter + Skew)) = Margin

**Table E-7 — S100 jitter budget (ns)**

| | Data jitter | Strobe jitter | Skew |
|---|---|---|---|
| Transmitter skew | | | 0.4 |
| Transmitter jitter | 0.80 | 0.80 | |
| Cable reflections | 0.13 | 0.13 | |
| Cable intersymbol | 0.1 | 0.1 | |
| Cable delay mismatch | | | 0.4 |
| Channel margin | 0.05 | 0.05 | |
| Jitter at receive pins | 1.08 | 1.08 | 0.8 |
| Receiver offset | 0.5 | 0.5 | 0.2 |

**Table E-7 — S100 jitter budget (ns) (*continued*)**

| | Data jitter | Strobe jitter | Skew |
|---|---|---|---|
| Receiver intersymbol and power supply rejection | 0.5 | 0.5 | |
| Flip flop setup and hold | 1.0 | 1.0 | |
| Total | 3.08 | 3.08 | 1.0 |

The margin for the S100 rate is equal to (10.17– (3.08 + 3.08 + 1.0) = 3.01 ns.

**Table E-8 — S200 jitter budget (ns)**

| | Data jitter | Strobe jitter | Skew |
|---|---|---|---|
| Transmitter skew | | | 0.15 |
| Transmitter jitter | 0.25 | 0.25 | |
| Cable reflections | 0.08 | 0.08 | |
| Cable intersymbol | 0.12 | 0.12 | |
| Cable delay mismatch | | | 0.4 |
| Channel margin | 0.05 | 0.05 | |
| Jitter at receive pins | 0.50 | 0.50 | 0.55 |
| Receiver offset | 0.25 | 0.25 | 0.1 |
| Receiver intersymbol and power supply rejection | 0.25 | 0.25 | |
| Flip flop setup and hold | 0.5 | 0.5 | |
| Total | 1.50 | 1.50 | 0.65 |

The margin for the S200 rate is equal to (5.08 – (1.50 + 1.50 + 0.65) = 1.48 ns.

**Table E-9 — S400 jitter budget (ns)**

| | Data jitter | Strobe jitter | Skew |
|---|---|---|---|
| Transmitter skew | | | 0.1 |
| Transmitter jitter | 0.15 | 0.15 | |
| Cable reflections | 0.035 | 0.035 | |
| Cable intersymbol | 0.13 | 0.13 | |
| Cable delay mismatch | | | 0.4 |
| Channel margin | 0 | 0 | |
| Jitter at receive pins | 0.315 | 0.315 | 0.50 |
| Receiver offset | 0.14 | 0.14 | 0.05 |
| Receiver intersymbol and power supply rejection | 0.1 | 0.1 | |
| Flip flop setup and hold | 0.2 | 0.2 | |
| Total | 0.755 | 0.755 | 0.55 |

The margin for the S400 rate is equal to (2.54 – (0.755 + 0.755 + 0565) = 0.48 ns.

## E.3 Cable PHY configuration example

This subclause gives a detailed example of the three phases of configuring a cable environment: bus initialization, tree identify, and self-identify.